```java
/**
package rr;

import java.text.SimpleDateFormat;
import java.util.concurrent.CyclicBarrier;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

/**
 * JAVA多线程实现龟兔赛跑
 *
 * 要求：1、兔子每秒跑5米，但是每10米要休息2秒 2、乌龟每秒钟4米，不休息 3、谁先到达终点，比赛结束
 *
 * @author renzenggang
 *
 */
public class RTRunner {

    // 测试
    public static void main(String[] args) {
        ExecutorService es = Executors.newFixedThreadPool(2);
        CyclicBarrier barrier = new CyclicBarrier(2, new Runnable() {
            @Override
            public void run() {
```

```
27                  SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
28                  System.out.println("参赛选手准备就绪；比赛开始！ " +
  sdf.format(System.currentTimeMillis()));
29              }
30          });
31          Rabbit rabbit = new Rabbit("小白兔", 5, es, barrier);
32          es.submit(rabbit);
33
34          Tortoise tortoise = new Tortoise("乌龟", 4, es, barrier);
35          es.submit(tortoise);
36
37          es.shutdown();
38      }
39
40      // 创建动物类：Animal
41      public static abstract class Animal implements Runnable {
42          protected String name;// 动物名称
43          protected int speed;// 动物速度
44          protected int now;// 当前已经跑的路程
45          protected ExecutorService executor;// 线程管理器，方便结束线程
46          protected CyclicBarrier barrier;// 等待计时器，要求两个参赛选手 都准备就绪后才   开始
  比赛
47          public static volatile boolean FINISH = false;// 是否比赛完成的标记
```

```
48          public final static int SUCCESS = 20;// 比赛的路程
49
50          Animal(String name, int speed, ExecutorService executor, CyclicBarrier
  barrier) {
51              this.name = name;
52              this.speed = speed;
53              this.executor = executor;
54              this.barrier = barrier;
55          }
56
57          Animal(String name, int speed, ExecutorService executor) {
58              this.name = name;
59              this.speed = speed;
60              this.executor = executor;
61          }
62
63          @Override
64          public abstract void run();
65
66          // 判断是否完成比赛
67          protected void finish() {
68              if (now >= SUCCESS) {
69                  System.out.println(name + " 跑完了,结束比赛!");
70                  FINISH = true;
```

```
71                  executor.shutdownNow();
72              }
73          }
74      }
75
76      // 创建小兔子
77      public static class Rabbit extends Animal {
78          Rabbit(String name, int speed, ExecutorService executor, CyclicBarrier
    barrier) {
79              super(name, speed, executor, barrier);
80          }
81
82          Rabbit(String name, int speed, ExecutorService executor) {
83              super(name, speed, executor);
84          }
85
86          @Override
87          public void run() {
88              try {
89                  barrier.await();// 等待参赛选手都准备就绪
90              } catch (Exception e1) {
91              }
92              SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
93              System.out.println(sdf.format(System.currentTimeMillis()));
```

```
 94              while (!FINISH) {
 95                  System.out.println(this.name + " : 我的速度是: " + this.speed + " 我跑了" + this.now + "米了");
 96                  try {
 97                      Thread.sleep(1000);
 98                      if (now % 10 == 0) {
 99                          Thread.sleep(1000);
100                      }
101                  } catch (InterruptedException e) {
102                  }
103                  now += speed;
104                  finish();
105              }
106          }
107      }
108
109      // 创建小乌龟
110      public static class Tortoise extends Animal {
111
112          Tortoise(String name, int speed, ExecutorService executor, CyclicBarrier barrier) {
113              super(name, speed, executor, barrier);
114          }
115
```

```
116         Tortoise(String name, int speed, ExecutorService executor) {
117             super(name, speed, executor);
118         }
119 
120         @Override
121         public void run() {
122             System.out.println();
123             try {
124                 Thread.sleep(2000);
125                 barrier.await();
126             } catch (Exception e1) {
127             }
128             SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
129             System.out.println(sdf.format(System.currentTimeMillis()));
130             while (!FINISH) {
131                 System.out.println(this.name + " : 我的速度是: " + this.speed + " 我跑
    了" + this.now + "米了");
132                 now += speed;
133                 finish();
134                 try {
135                     Thread.sleep(1000);
136                 } catch (InterruptedException e) {
137                 }
138             }
```

```
139         }
140     }
141 }
142
```